활 동 보 고 서

회차	4회차	지도교수 확인란	
일시	2019.8.14.(수) 15:00~18:00		
장소	충남대학교 도서관 그룹 스터디룸		
주제	파이썬과 루비(3) 객체제작부터 패키지 매니저까지.		

학습 내용 요약

<객체 지향 프로그래밍(Object Oriented Programming)>

객체는 OOP의 기반이 되는 개념중의 하나이다. 이것은 함수만 묶어 놓았던 모듈과는 다르게, 객체에서는 변수와 함수를 묶어서 찍어낼 수 있는 개념이다. 이렇게 객체를 기반으로 생성된 것을 인스턴스라고 한다.

인스턴스가 생성될 때 initialize라는 메소드(생성자)를 사용하면 객체를 만들 때 초기화가 된다. 이는 자동차에 시동을 거는 것과 같다. 변수 앞에 골뱅이(@)가 붙이면 그것을 인스턴스 변수라고 한다. 이는 인스턴스 내에서 사용할 수 있는 기호가 된다.

각각의 인스턴스가 서로 다른 데이터를 가지고있다!
= 서로 다른 상태를 가지고 있고, 그 상태에 따라서 메소드를 실행한 결과가 달라진다.
=메소드의 동작방법이 달라진다
=행위가 달라진다(같은 행위지만 다른 결과를 만들어낸다)
루비에서는 객체가 대문자로 시작해야 함
클래스가 시작되는 것은 end로 끝내줘야 함
객체를 사용할 수 있도록 만드는 입장에서 객체란 무엇인가를 살펴봄.
-

<클래스와 변수>

클래스: 프로그램을 만드는 과정에서는 여러가지 변수와 함수들이 있음. 그것들 사이에서 연관되어있는 변수와 함수를 그룹핑해서 클래스라고 하는 일종의 수납공간에 정리정돈해 놓는 느낌.

다음에는 class 키워드를 사용해서 객체를 만들어 준다. 이때, 루비는 역시 end 키워드를 사용해서 객체 생성 코드를 닫아주며, 파이썬은 class뒤에 객체이름(object):를 적어 준다.

왜 변수와 함수?
- 프로그램에서 어떠한 정보를 저장하는 그룹= 변수
연관되어있는 각각의 로직들을 그룹핑 해놓은 그릇 = 일종의 함수
변수& 함수 다시 모아서 그룹핑& 패키징
=> 클래스

이렇게 만들어진 클래스 자체를 우리가 직접 사용하는 것보다는 이러한 클래스를 복제해서 클래스와 톡같은 인스턴스들을 만듦 > 따라서 이러한 인스턴스들에는 변수와 함수가 모두 들어있음.

*인스턴스마다 변수(실제로 저장되어있는 값이, 데이터가 다르고, 그 데이터가 다름에 따라 각각의 함수가 동작하는 방법이 달라짐)
객체지향 프로그래밍에서는 함수라는 말 대신에 메소드 라는 말을 씀.
변수라는 말 대신 적성 필드 또는 상태라는 말도 씀. 상태(=변수), 행위(=메소드를 의미하는 느낌)

파이썬은 메소드 바깥(외수)에서 인스턴스 변수로 접근하는 게 허용이 되지만 루비는 허용하지 않음(에러)

<get, set 메소드>

getValue: '인스턴스변수 Value의 값을 가져오는 메소드이다'
setValue:'인스턴스변수 Value의 값을 설정하는 메소드이다'

인스턴스 변수를 직접수정하지 않고 메소드를 통해 수정하는 것을 set, get 메소드라고 함.

파이썬은 인스턴스 변수를 메소드 바깥에서도 얼마든지 수정할 수 있지만 루비는 허용하지 않음.(루비에선 인스턴스에 직접 접근할 수 있는 방법을 제공하지 않는 셈)

따라서 루비에서는 set/get메소드를 적극 사용하기를 권장.
객체를 만들 때 초기화 해야 하는 어떤 작업들을 initialize라고 하는 약속되어 있는 메소드의 본문에 넣어두면 루비가 initialize를 자동으로 추출하기 때문에 우리는 그것이 실행될 것이라는 것을 알 수가 있을 것

인스턴스가 만들어질 때 이니셜라이즈라고 하는 생성자가 실행이 됨. 인스턴스가 생성될 때 실행되기를 기대하는 코드를 이니셜라이즈 함수의 본체에 넣어두면 그것이 실행되게 될 것.

클래스가 인스턴스화化될 때 실행되도록 약속되어 있는 초기화와 관련되어있는 것이 생성자 메소드였고, 루비에선 이니셜라이즈라는 이름을 사용함
생성자 함수, 생성자 메소드가 실행될 때 함수 뒤에 있는 값이 전달이 돼서, 우리가 코드를 실행해서 v1과 v2라는 변수에 담겨있는 값을 출력할 수 있게 됨.

<오버라이드(Override)>

오버라이드는 재정의라는 의미를 내포하고 있다. 상속이라는 개념에서 상당히 중요한 기능이며, 복잡해진 객체 지향을 좀 더 잘 사용하기 위해 만들어진 것이다.

오버라이드는 상속 받은 메소드를 재정의 하는 것을 말한다. 부모 객체에서 자식 객체로 메소드가 넘어갈 때 이 메소드의 기능을 수정 하기 위해 해당 메소드의 코드를 다시 작성하는데 이것이 오버라이드이다.

<다중상속(Multiple Inheritance)>

기존에 부모 객체로 부터 메소드를 받을 수 있는 것을 상속이라고 하며, 여러개의 부모객체로 부터 상속을 받는 것을 다중 상속이라고 한다.

객체 지향을 하는 언어들이 모두 다중 상속을 지원 하는 것은 아니며, 대부분은 지원하지 않는다. 이유는 죽음의 다이아몬드라는 다중상속의 크나큰 단점 때문인데, 이 때문인지 몰라도 루비에서는 다중상속을 지원하지 않고 파이썬에서는 지원한다.
대신 루비에서는 Mixin이라는 기능을 이용해서 비슷한 목적을 이룰수 있음.

<믹스인(Mixin)>

앞서 파이썬에서 다중 상속에 대해 살펴보았다. 다중 상속에서 루비에 대해 다루지 않은 이유는 루비는 다중상속을 지원하지 않기 때문이다. 대신 믹스인이라는 세련된 방법을 사용한다. 믹스인이라는 것은 객체와 모듈의 관계이다.

학습 방법 및 과정

상속이라는 개념이 이해가 안 되어서 다같이 고민하고 있을 때, 김현지 학우가 생활코딩을 듣고 자신이 이해한 바를 설명해 주었다. 설명은 다음과 같다.
"우리가 자전거를 만든다고 했을 때, 부품들을 조합해서 만들게 되는데, 이렇게 사용되는 부품을 함수라고 생각해봤을 때 함수라는 부품을 조합해서 자전거라는 객체를 만들 수 있잖아요. 이 자전거를 다른 사람에게 팔았는데, 팔리고 나서 새로운 기능을 달고 싶어 한다고 쳐봐요. 그러면 자전거에 전조등 같은 걸 달게 되었는데, 처음에 깔끔하게 자전거 기능만 담고 있던 자전거에 전조등의 기능을 추가하면서 새로운 객체가 된 거죠?. 위의 예처럼 새로운 기능을 추가해서 새로운 객체를 만드는 것. 이것을 상속(Inheritance)이라고 하는 거래요."

또한 지금까지 배운 변수와 메소드는 '인스턴스 멤버'였다. 이번 시간에 배운 내용은 '클래스'에 소속되어있는 변수와 메소드였는데 인스턴스의 멤버와 클래스멤버의 차이점과, 왜 이런 차이점을 가지고 있는지에 대해 생각해보는 시간을 가졌다.

파이썬과 루비에서 인스턴스 변수로 변환하는? 방법에 대해서도, 중요한 부분이니 함께 이야기 해보기로 했다. 다음은 함께 이야기했던 부분을 요약한 것이다.

"루비는 그 변수 이름 앞에 @를 붙이면 인스턴스 에 속해있는 인스턴스 변수가 되는 반면
파이썬에서는 메소드의 첫번째 인자가 인스턴스라는 것. 인스턴스가 첫번째 인자임을 가리키는 것이기 때문에 앞에 첫번쨰 인자를 적어줘야 함."

다음으로, 학습 내용 중에서 각자가 중요하다고 생각하는 점을 하나씩 꼽아보았다.

신예슬 학우는 루비에서 "지역변수"는 해당 메소드 아래에서만 사용할 수 있다"는 점을,

강미규 학우는 "파이썬에는 앞에 self.를 붙여줘야 함, 파이썬에서의 메소드들은 첫번째 매개변수를 꼭 정의해줘야 한다"는 것을,

이정란 학우는 "인스턴스가 생성될 때 initialize라는 메소드(생성자)를 사용하면 객체를 만들 때 초기화가 된다."는 것을,

김예린 학우는 "프로그램에서 어떠한 정보를 저장하는 그릅을 변수라고 하며, 연관되어있는 각각의 로직들을 그룹핑해놓은 그릇을 일종의 함수라고 한다" 는 점을,

김현지 학우는 "get,set메소드"를 이번 주차의 핵심 내용으로 꼽았다.

<학습 소감>

정란 학습소감: 생활코딩 강의가 일상생활에서 바탕된 소재라는 것에서 너무 맘에 들었고, 우리가 실제 쓰는 웹이 어떻게 만들어지고, 무엇을 전하는지의 궁금증이 해소되었다. 아직 갈길이 멀지만 아예 시작조차 하지 않거나 도중에 포기하기보다는 어려워도 매일 한 걸음씩 걸어가야겠다는 생각이 들었다.
앞으로의 계획
1. 세상과 소프트웨어가 접목한다면 어떻게 변할지 창의적으로 바라볼 수 있는 눈을 심고 싶다.
2. 답이 정해져 있는 풀이 식이 아닌 자신의 논리대로 문제를 해결할 수 있는 능력을 키우고 싶다.
3. 이 강의 학습을 통해 미래 사회에 꼭 필요한 지식들을 체득할 수 있게 하고자 한다.

예슬 학습소감:
저번 시간에 객체지향 프로그래밍에 대해 처음 배웠을 때는 강의를 다 보고 개념을 이해를 했어도 깊게 와닿지는 않았다. 하지만 이번 시간 강의를 끝까지 완강하고 나니, 객체를 만들어보고 다양한 활용과 객체를 사용하는 이유에 대해서도 배우고 나니 좀 더 명확하게 이해할 수 있었다. 수업 마지막에는 약간 번외편처럼 패키지 매니저를 이용한 크롤링도 해보았는데, 평소 관심있는 분야라 재밌었고 앞으로 다른 웹페이지에서도 크롤링을 해보고싶다.

Python & Ruby를 공부하며 느낀점 :
처음 배우는 언어를 파이썬과 루비로 정하길 잘했다는 생각이 든다. 객체지향 언어를 다른 언어들에 비해 복잡하거나 어렵지 않게 배울 수 있다는 점과 입문자에게 접근이 쉬운 언어이기 때문이다. 파이썬과 루비로 컴퓨터 언어를 대략 맛보고 나니 첫 언어를 JAVA나 C언어로 시작했다면 많이 어려웠을 것 같다는 생각이 든다. 강의가 입문자와 비전공자에게 맞춰진 강의라 더욱 세세한 설명과 코딩 실습 위주로 되어있기 때문에, 배우는 과정이 길고 예제도 많았다. 그만큼 힘들고 가끔 지치기도 했지만 언어에 대한 탄탄한 기본기를 갖출 수 있었다. 다 끝나고 나니 두 가지 언어를 알게되었고, 다른 언어를 배울 이해력을 가지게 되어서 너무 뿌듯하다. 방학 때 단 하나의 언어라도 배우고 싶었는데 그 목표를 이룬 것 같아서 보람차다. 파이썬을 알기 전과 후가 많이 다른 것 같다. HTML/CSS로 어느정도 컴퓨터의 동작원리와 역사는 알게되었지만 그래도 아직 비전공자에서 못벗어 나는 것 같았는데, 파이썬을 알고 나니 컴퓨터의 세계로 한 발 내딛은 느낌이 든다. 파이썬과 루비를 함께 배웠지만 앞으로는 파이썬에 좀 더 주력할 계획을 세웠다. 앞으로 배우게 될 JAVA도 열심히 배워서 두 가

지 언어를 자유자재로 활용하는 실력자가 되고 싶다.

앞으로의 활용 계획 공유 :
파이썬은 활용도가 무궁무진한 언어이고 강력한 언어이다. 데이터 크롤링 등을 활용해 소위 말하는 빅데이터 분야에서도 활용 가능하며 머신러닝을 배우면서 텐서플로우를 사용할 때 활용되는 언어이기 때문이다. 앞으로 이 기본기를 바탕으로 간단한 게임제작이나 크롤링, 텐서플로우 등을 사용하며 언어를 더욱 익힐 예정이다. 사실 파이썬과 루비를 함께 배우면서 이해에 더 도움되는 점도 있었지만 파이썬을 익히기에는 오히려 헷갈렸기 때문에 다른 파이썬 교재나 강의를 한번 더 들으
면서 손에 익을 때까지 좀 더 깊게 공부해보고싶다.

미규: 파이썬과 루비는 html과 다르게 뭔가 바로바로 보이는 것이 없어서 배울 때 힘이 좀 빠지긴 했지만, 그래도 한 줄 한 줄 출력이 되는 것이 조금은 뿌듯한 것 같기도 했다. 사실 파이썬과 자바의 객체지향 프로그래밍도 어려웠는데, 자바는 아예 객체지향 프로그래밍 기반이라고 해서 걱정이 되는 것도 사실이다. 하지만 그래도 파이썬과 루비를 공부하듯이 하면 자바도 어렵지만 잘 해낼 수 있을 것이란 생각이 든다. 다만 아직 파이썬으로 뭘 할 수 있는진 구체적으로 와닿지가 않아서 앞으로 파이썬으로 뭘 할 수 있을 지에 대해 고민이 많다.

앞으로의 학습 계획: 우선은 파이썬에 대한 이해가 가장 먼저일 것 같다. 가장 간단하다고들 하고, 또 활용도가 높은 언어라고들 하니까 우선은 파이썬의 기초를 한 번 더 다지고 나서 다른 언어(다음 주차부터 학습하게 될 자바) 학습에 들어가야 할 것이다.

현지: 들어보기만 했던 객체 지향 프로그래밍을 배웠다. 일단은 이론을 배우는 단계였기 때문에 이것으로 무엇을 할 수 있을지 잘 모르겠다. 아직은 막연한 개념이다.
함수를 배울 때와 느낌이 다르다. 더 큰 프로젝트를 만드는 데 필요한 것들을 배운다는 느낌이 든다. 하지만 이것도 엄청 기초 단계이니 어렵지만 이해하려고 애썼다. 객체 지향 프로그래밍부터 복잡해져서 일단 끝까지 다 듣고 몇 번 반복해서 들어보기로 했다. 그런데 그것만으로는 충분하지 않을 것 같아서 나는 다른 파이썬 교재도 병행하며 개념을 익히거나 여러 파이썬 예제를 찾아볼 생각이다.

예린: 파이썬이 사실 쉬운 언어라고 해서 만만히 봤는데 역시 쉬운 게 없는 것 같다... 특히 조건문과 반복문에서 정말 많은 좌절을 겪었다. 코드를 맞게 입력했는데도 자꾸 에러가 났다. 동아리원들과 이야기를 하면서 해결할 수 있는 시간이 정말 값지다고 느꼈고, 이래서 코딩에 있어서 협업이 중요하다고 하는 거구나 싶었다. 또 파이썬과 루비를 함께 배우니 두가지 언어를 한번에 배울 수 있다는 장점도 있었지만, 파이썬의 문법과 루비의 문법이 비슷하지만 다른 경우(elsif와 elif)에는 헷갈리기도 하고, 그 탓에 잘못 입력하여 에러가 나는 경우도 잦았다. 둘을 잊지 않도록 꾸준히 복습해야겠다고 느꼈다.

앞으로의 계획:
1. '생활코딩' 강좌에서 미처 짚고 넘어가지 못한 부분이나 에러가 쉽게 나던 부분을 중심으로 다시 천천히 학습하려 한다.
2. '파이썬'의 기초를 확실히 다지고 자바나 다른 프로그래밍 언어와의 차이점을 비교해 본다.
3. 다른 프로그래밍 언어와의 차이점을 알았다면, 내게 가장 적합한 프로그래밍 언어, 혹은 내가 진출하려고 하는 직군, 혹은 하려 하는 작업에 걸맞는 것을 찾아보려 한다.

<table>
<tr><th colspan="4">활동 사진 첨부</th></tr>
<tr>
<td>학습활동①</td>
<td>  </td>
<td>학습결과물①</td>
<td>

```ruby
# ruby
module M1
  def m1_m
    p "m1_m"
  end
end
module M2
  def m2_m
    p "m2_m"
  end
end
class C
  include M1, M2
end
c = C.new()
c.m1_m()
c.m2_m()
```

믹스인에 대한 실습(이정란 학우)
</td>
</tr>
<tr>
<td>학습활동②</td>
<td>  </td>
<td>학습결과물②</td>
<td>

```python
#생성자(constructor)
#메소드를 정의할 때는 def로
class Cal(object):
    def __init__(yelyn, v1,v2):
        yelyn.v1 = v1
        yelyn.v2 = v2

    def add(yelyn):
        return yelyn.v1+yelyn.v2

    def subtract(yelyn):
        return yelyn.v1-yelyn.v2

c1 = Cal(10,10)
print(c1.add())
print(c1.subtract())
c2 = Cal(30,20)
print(c2.add())
print(c2.subtract())
```

```
20
0
50
10
[Finished in 0.103s]
```

김예린 학우의 조건문(파이썬,아톰)실행화면. self대신 yelyn이라는 연산자를 만들어 사용했다.
</td>
</tr>
</table>